Java

Generics

Uso di Generics nelle API di Java

```
// Removes 4-letter words from c. Elements must be strings
  static void expurgate(Collection c) {
    for (Iterator i = c.iterator(); i.hasNext(); )
      if (((String) i.next()).length() == 4)
        i.remove();
  }
```

Problemi?

Uso di Generics nelle API di Java

```
// Removes 4-letter words from c. Elements must be strings
  static void expurgate(Collection c) {
    for (Iterator i = c.iterator(); i.hasNext(); )
      if (((String) i.next()).length() == 4)
        i.remove();
  }
```

Here is the same example modified to use generics:

```
  // Removes the 4-letter words from c
  static void expurgate(Collection<String> c) {
    for (Iterator<String> i = c.iterator(); i.hasNext(); )
      if (i.next().length() == 4)
        i.remove();
  }
```

In Java 5
molte classi
sono state
riscritte
usando i
generics

Generic Pila

```
public class Pila <T> {
  …
  public Pila () {…   }
  private void cresci(int dim) {…}

  public final void inserisci(T k) {
    if (marker == size) {
      cresci(defaultGrowthSize);
    }
    contenuto[marker] = k;
    marker++;
  }

  public T estrai() {
    assert(marker > 0):"Estrazione da Pila vuota";
    return (T) contenuto[--marker];
  }
```

Generic Pila

```
public static void main(String args[]) {
    int dim = 10;
    Pila<Integer> s = new Pila<Integer>();
    for (int k = 0; k < dim; k++) {
      s.inserisci(new Integer(k));
    }
    for (int k = 0; k < 3 * dim; k++) {
      Integer w = s.estrai();
      // Integer w = (Integer) s.estrai();
      System.out.println(w);
    }
  }

}
```

Generic Pila

```
public static void main(String args[]) {
    int dim = 10;
    Pila<Integer> s = new Pila<Integer>();
    for (int k = 0; k < dim; k++) {
      s.inserisci(new String("pippo"));
    }
    for (int k = 0; k < 3 * dim; k++) {
      Integer w = s.estrai();
     // Integer w = (Integer) s.estrai();
      System.out.println(w);
    }
  }
}
```

```
Pila.java:43: inserisci(java.lang.Integer)
in Pila<java.lang.Integer> cannot be
applied to (java.lang.String)
      s.inserisci(new String("pippo"));
       ^
```

Pbm: Generics are not usable…

- **for creation of arrays**
- **in an instanceof expression**

Generic Pila

```
public class Pila <T> {
  …
  public Pila () {…   }
  private void cresci(int dim) {…}

  public final void inserisci(T k) {
    if (marker == size) {
      cresci(defaultGrowthSize);
    }
    contenuto[marker] = k;
    marker++;
  }

  public T estrai() {
    assert(marker > 0):"Estrazione da Pila vuota";
    return (T) contenuto[--marker];
  }
```

Note: Pila.java
uses unchecked or
unsafe operations.

Definizione

A generic type is a reference type that has one or more type parameters. In the definition of the generic type, the type parameter section follows the type name. It is a comma separated list of identifiers and is delimited by angle brackets.

```
class Pair<X,Y>  {
  private X first;
  private Y second;   public Pair(X a1, Y a2) {
    first  = a1;
    second = a2;
  }
  public X getFirst()  { return first; }
  public Y getSecond() { return second; }
  public void setFirst(X arg)  { first = arg; }
  public void setSecond(Y arg) { second = arg; }
}
```

Definizione - continua

The class Pair has two type parameters X and Y .

They are replaced by type arguments when the generic type Pair is instantiated.

For instance, in the declaration Pair<String, Date> the type parameter X is replaced by the type argument String and Y is replaced by Date .

The scope of the identifiers X and Y is the entire definition of the class. In this scope the two type parameters X and Y are used like they were types (with some restrictions).

Esempio

```
public void printPair( Pair<String,Long> pair) {
    System.out.println("("+pair.getFirst()+",“
             +pair.getSecond()+")");
  }

  Pair<String,Long> limit =
         new Pair<String,Long> ("maximum",1024L);
  printPair(limit);
```

Wildcard instantiation

```
public void printPair( Pair<?,?> pair) {
     System.out.println("("+pair.getFirst()+",“
                 +pair.getSecond()+")");
  }

  Pair<?,?> limit =
          new Pair<String,Long> ("maximum",1024L);
  printPair(limit);
```

Referenze su generics:

- **Il meglio:**

- **http://www.angelikalanger.com/GenericsFAQ/JavaGenericsFAQ.html**

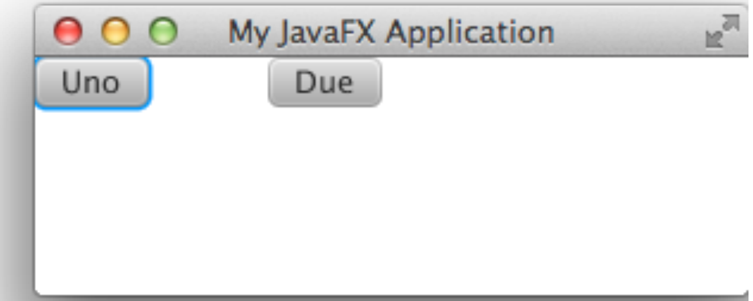
Autoboxing

Autoboxing/Autounboxing

```
public static void main(String args[]) {
   int dim=10;
   Pila<Integer> s=new Pila(); // s= new Coda();
   for (int k=0;k<dim;k++){
   //Integer o=new Integer(k);
   //s.inserisci(o);
   s.inserisci(k);
   }
   for (int k=0;k<3*dim;k++) {
     //int j= Integer.parseInt(s.estrai());
     int j= s.estrai();
     System.out.println(j);
   }
}
```

Eventi di tastiera: il fuoco

Un app con due bottoni…



```
public class Keyboard1 extends Application {
    int counter=0;
    public void start(Stage stage) {
        TilePane box=new TilePane();
        box.setHgap(50);
        final Button b1=new Button("Uno");
        final Button b2=new Button("Due");
        box.getChildren().addAll(b1,b2);
        EventHandler actionHandler=new EventHandler(){
            public void handle(Event t) {
                System.out.println((counter++)+
                        ((Button)(t.getTarget())).getText());
            }
        };
        b1.addEventHandler(ActionEvent.ACTION, actionHandler);
        b2.addEventHandler(ActionEvent.ACTION, actionHandler);
```

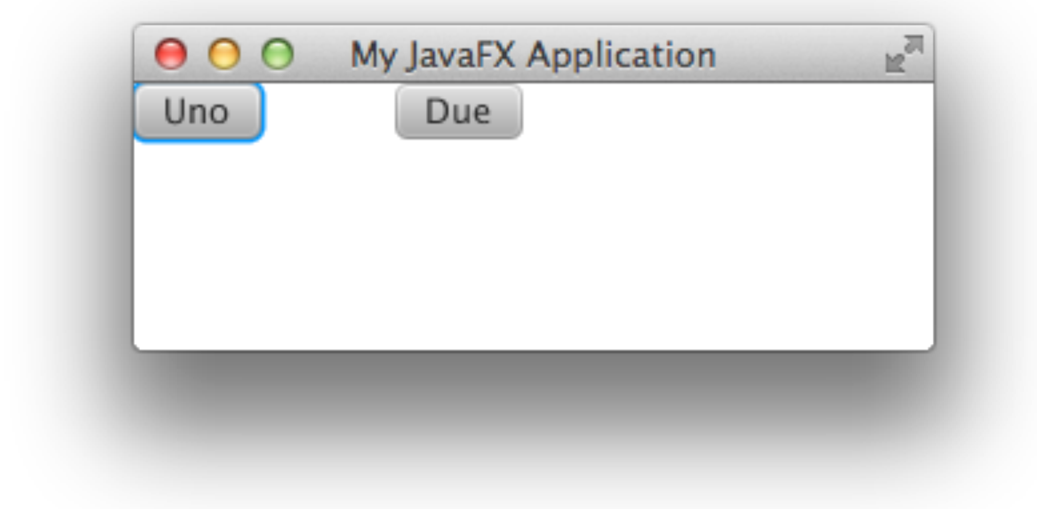
```
0Uno
1Uno
2Uno
3Uno
```

…che cattura gli eventi di keyboard

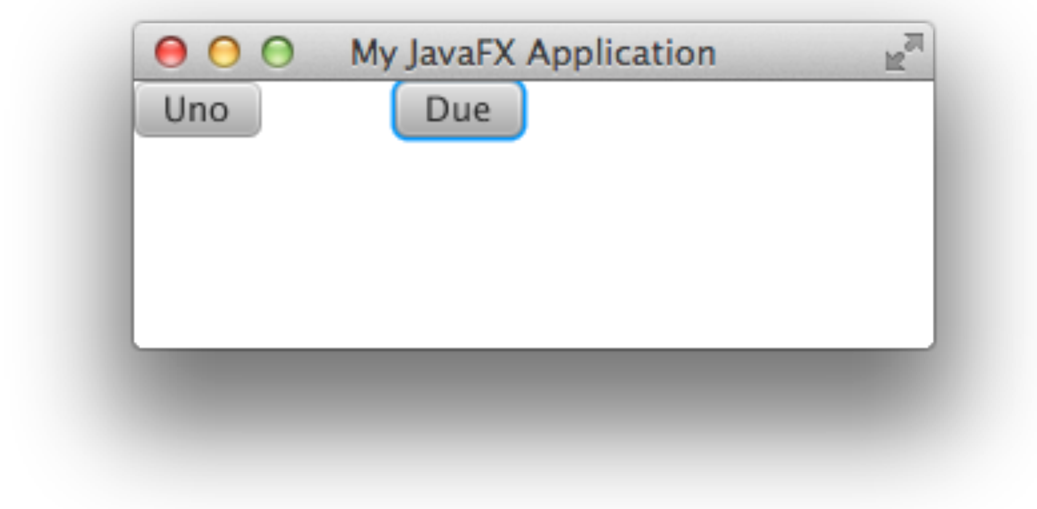
```
        EventHandler<KeyEvent> keyEventHandler =
                    new EventHandler<KeyEvent>() {
                        public void handle(KeyEvent keyEvent) {
                            if (keyEvent.getCode() == KeyCode.U) {
                                b1.fireEvent(new ActionEvent());
                                System.out.println(keyEvent.getSource()
                                     +" => "+keyEvent.getTarget());
                            }
                        }
                    };
         Scene scene = new Scene(box, 400, 300);
         b1.addEventHandler(KeyEvent.KEY_PRESSED, keyEventHandler);
         stage.setTitle("My JavaFX Application");
         stage.setScene(scene);
         stage.show();
     }
       public static void main(String[] args) {
         Application.launch(args);
  }
}
```

```
Button[id=null, styleClass=button] => Button[id=null, styleClass=button]
```

ma funziona?



SI!



NO!

Sistemiamola.

```
   EventHandler<KeyEvent> keyEventHandler =
               new EventHandler<KeyEvent>() {
                   public void handle(KeyEvent keyEvent) {
                       if (keyEvent.getCode() == KeyCode.U) {
                           b1.fireEvent(new ActionEvent());
                           System.out.println(keyEvent.getSource()
                                 +" => "+keyEvent.getTarget());
                       }
                   }
               };
    Scene scene = new Scene(box, 400, 300);
    //b1.addEventHandler(KeyEvent.KEY_PRESSED, keyEventHandler);
    stage.addEventHandler(KeyEvent.KEY_PRESSED, keyEventHandler);
    stage.setTitle("My JavaFX Application");
    stage.setScene(scene);
    stage.show();
}
```

```
javafx.scene.Scene@68a08ca7 => Button[id=null, styleClass=button]
```

Ora gestiamo anche l'altro bottone.

```
EventHandler<KeyEvent> keyEventHandler =
                   new EventHandler<KeyEvent>() {
                       public void handle(final KeyEvent keyEvent) {
                           System.out.println(keyEvent.getSource()+"
                                 => "+keyEvent.getTarget());
                           switch (keyEvent.getCode()){
                               case U:
                               case DIGIT1:
                                   b1.fireEvent(new ActionEvent());
                                   break;
                               case D:
                               case DIGIT2:
                                   b2.fireEvent(new ActionEvent());
                                   break;
                           }
                       }
                   };
```

Event Chain

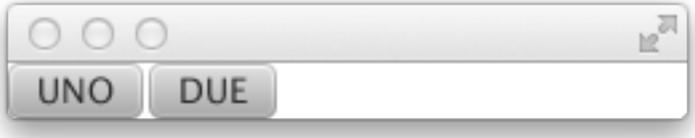
Events

User Action	Event Type	Class
Key on the keyboard is pressed.	`KeyEvent`	`Node, Scene`
Mouse is moved or a button on the mouse is pressed.	`MouseEvent`	`Node, Scene`
Full mouse press-drag-release action is performed.	`MouseDragEvent`	`Node, Scene`
Input from an alternate method for entering characters (typically for a foreign language) is generated, changed, removed, or committed.	`InputMethodEvent`	`Node, Scene`
Platform-supported drag and drop action is performed.	`DragEvent`	`Node, Scene`
Object is scrolled.	`ScrollEvent`	`Node, Scene`
Rotation gesture is performed on an object	`RotateEvent`	`Node, Scene`
Swipe gesture is performed on an object	`SwipeEvent`	`Node, Scene`
An object is touched	`TouchEvent`	`Node, Scene`
Zoom gesture is performed on an object	`ZoomEvent`	`Node, Scene`
Zoom gesture is performed on an object	`ZoomEvent`	`Node, Scene`
Context menu is requested	`ContextMenuEvent`	`Node, Scene`
Button is pressed, combo box is shown or hidden, or a menu item is selected.	`ActionEvent`	`ButtonBase,ComboBoxBase,ContextMenu,MenuItem, TextField`

Item in a list, table, or tree is edited.	`ListView.EditEvent`
	`TableColumn.CellEditEvent`
	`TreeView.EditEvent`
		`ListView`
		`TableColumn`
		`TreeView`

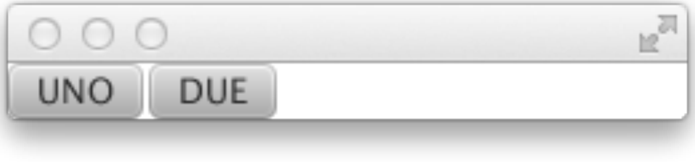
Media player encounters an error.	`MediaErrorEvent`	`MediaView`
Menu is either shown or hidden.	`Event`	`Menu`
Popup window is hidden.	`Event`	`PopupWindow`
Tab is selected or closed.	`Event`	`Tab`
Window is closed, shown, or hidden.	`WindowEvent`	`Window`

Event chain v.1- 1



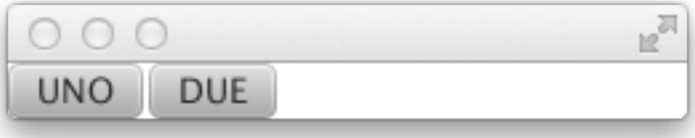
```
public class EventFilterDemo extends Application {
   public void start(final Stage stage) {
      EventHandler handler=new EventHandler<ActionEvent>() {
         @Override
         public void handle(ActionEvent t) {
            EventTarget target=t.getTarget();
            Object source=t.getSource();
            String id=null;
            if (source instanceof Node {
                id=((Node)source).getId();
            } else if (source instanceof Stage) {
                id="STAGE";
            } else if (source instanceof Scene) {
                id="SCENE";
            } else {
                System.out.println("Unrecognized Object"+source);
            }
            System.out.println("HANDLER:"+id+" "+source+" ==> "
                                             +target);
         }
      };
```

Event chain v.1- 2



```
EventHandler filter=new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent t) {
        EventTarget target=t.getTarget();
        Object source=t.getSource();
        String id=null;
        if (source instanceof Node {
            id=((Node)source).getId();
        } else if (source instanceof Stage) {
            id="STAGE";
        } else if (source instanceof Scene) {
            id="SCENE";
        } else {
            System.out.println("Unrecognized Object"+source);
        }
        System.out.println("FILTER:"+id+" "+source+" ==> "
                                                +target);
    }
};
```

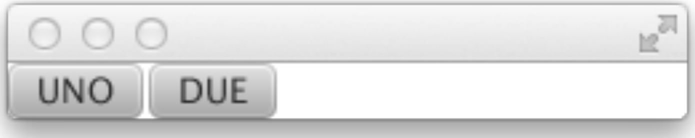
Event chain v.1- 3



```
TilePane layout=new TilePane();
Button button=new Button("Uno");
Button button2=new Button("DUE");
layout.getChildren().addAll(button,button2);
Scene scene = new Scene(layout);
layout.setId("STACKPANE");
button.setId("BUTTON");
button2.setId("BUTTON2");
scene.addEventFilter(ActionEvent.ACTION,filter);
scene.addEventHandler(ActionEvent.ACTION,handler);
stage.addEventFilter(ActionEvent.ACTION,filter);
stage.addEventHandler(ActionEvent.ACTION,handler);
layout.addEventFilter(ActionEvent.ACTION,filter);
layout.addEventHandler(ActionEvent.ACTION,handler);
button2.addEventFilter(ActionEvent.ACTION,filter);
button2.addEventHandler(ActionEvent.ACTION,handler);
button.addEventFilter(ActionEvent.ACTION,filter);
button.addEventHandler(ActionEvent.ACTION,handler);
stage.setScene(scene);
stage.show();
}
```

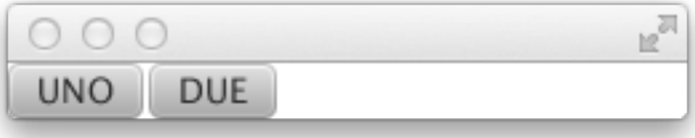
```
public static void main(String[] args) {
      Application.launch(args);
  }
```

Output



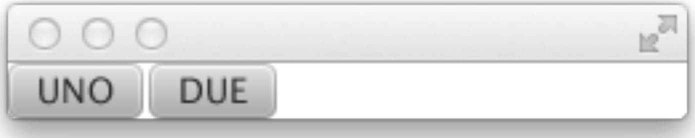
```
FILTER:STAGE javafx.stage.Stage@7c1031ba ==> Button[id=BUTTON-1, styleClass=button]
FILTER:SCENE javafx.scene.Scene@b30e9f8 ==> Button[id=BUTTON-1, styleClass=button]
FILTER:STACKPANE TilePane[id=STACKPANE, styleClass=root] ==> Button[id=BUTTON-1, styleClass=button]
FILTER:BUTTON-1 Button[id=BUTTON-1, styleClass=button] ==> Button[id=BUTTON-1, styleClass=button]
HANDLER:BUTTON-1 Button[id=BUTTON-1, styleClass=button] ==> Button[id=BUTTON-1, styleClass=button]
HANDLER:STACKPANE TilePane[id=STACKPANE, styleClass=root] ==> Button[id=BUTTON-1, styleClass=button]
HANDLER:SCENE javafx.scene.Scene@b30e9f8 ==> Button[id=BUTTON-1, styleClass=button]
HANDLER:STAGE javafx.stage.Stage@7c1031ba ==> Button[id=BUTTON-1, styleClass=button]
FILTER:STAGE javafx.stage.Stage@7c1031ba ==> Button[id=BUTTON-2, styleClass=button]
FILTER:SCENE javafx.scene.Scene@b30e9f8 ==> Button[id=BUTTON-2, styleClass=button]
FILTER:STACKPANE TilePane[id=STACKPANE, styleClass=root] ==> Button[id=BUTTON-2, styleClass=button]
FILTER:BUTTON-2 Button[id=BUTTON-2, styleClass=button] ==> Button[id=BUTTON-2, styleClass=button]
HANDLER:BUTTON-2Button[id=BUTTON-2, styleClass=button] ==> Button[id=BUTTON-2, styleClass=button]
HANDLER:STACKPANE TilePane[id=STACKPANE, styleClass=root] ==> Button[id=BUTTON-2, styleClass=button]
HANDLER:SCENE javafx.scene.Scene@b30e9f8 ==> Button[id=BUTTON-2, styleClass=button]
HANDLER:STAGE javafx.stage.Stage@7c1031ba ==> Button[id=BUTTON-2, styleClass=button]
```

Event chain v.2 - 1



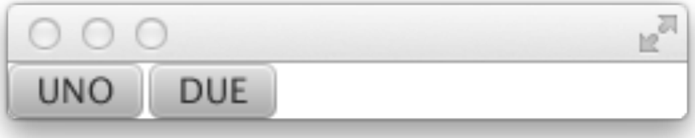
```
public void start(final Stage stage) {
        class SuperHandler<T extends Event> implements EventHandler<T>{
            SuperHandler() {  super(); }
            protected EventTarget target;
            protected Object source;
            protected String id;
            @Override
            public void handle(T t) {
                target=t.getTarget();
                source=t.getSource();
                id=null;
                if (source instanceof Node) {
                    id=((Node)source).getId();
                } else if (source instanceof Stage) {
                    id="STAGE";
                } else if (source instanceof Scene) {
                    id="SCENE";
                } else {
                    System.out.println("Unrecognized Object"+source);
                }
        };
```

Event chain v.2 – 2



```
SuperHandler<ActionEvent> filter=new SuperHandler<ActionEvent>() {
  public void handle(ActionEvent t) {
     super.handle(t);
     System.out.println("FILTER:"+id+" "+source+" ==> "+target);
  }
};
SuperHandler<ActionEvent> handler=new SuperHandler<ActionEvent>() {
   public void handle(ActionEvent t) {
     super.handle(t);
     System.out.println("HANDLER:"+id+" "+source+" ==> "+target);
  }
};
```

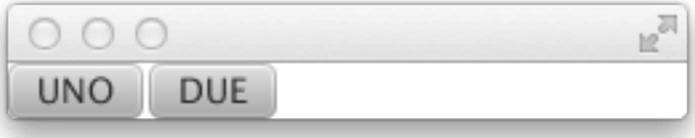
Event chain – cutter 1



```
 SuperHandler<ActionEvent> filter=new SuperHandler<ActionEvent>() {
    public void handle(ActionEvent t) {
       super.handle(t);
       System.out.println("FILTER:"+id+" "+source+" ==> "+target);
    }
 };
 SuperHandler<ActionEvent> handler=new SuperHandler<ActionEvent>() {
     public void handle(ActionEvent t) {
       super.handle(t);
       System.out.println("HANDLER:"+id+" "+source+" ==> "+target);
    }
 };

SuperHandler<ActionEvent> cutter=new SuperHandler<ActionEvent>() {
     public void handle(ActionEvent t) {
       super.handle(t);
       System.out.println("CUTTER:"+id+" "+source+" ==> "+target);
       t.consume();
     }
};
```

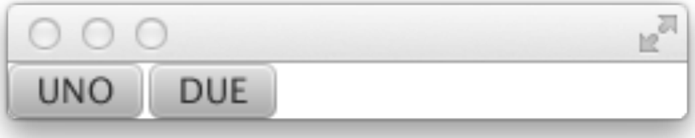
Event chain – cutter 2a



```
scene.addEventFilter(ActionEvent.ACTION,filter);
scene.addEventHandler(ActionEvent.ACTION,handler);
stage.addEventFilter(ActionEvent.ACTION,filter);
stage.addEventHandler(ActionEvent.ACTION,handler);
layout.addEventFilter(ActionEvent.ACTION,cutter);
layout.addEventHandler(ActionEvent.ACTION,handler);
button.addEventFilter(ActionEvent.ACTION,cutter);
button.addEventHandler(ActionEvent.ACTION,handler);
```

```
FILTER:STAGE javafx.stage.Stage@7c1031ba ==> Button[id=BUTTON-1, styleClass=button]
FILTER:SCENE javafx.scene.Scene@b30e9f8 ==> Button[id=BUTTON-1, styleClass=button]
CUTTER:STACKPANE TilePane[id=STACKPANE, styleClass=root] ==> Button[id=BUTTON-1, styleClass=button]
```

Event chain – cutter 2b



```
scene.addEventFilter(ActionEvent.ACTION,filter);
scene.addEventHandler(ActionEvent.ACTION,handler);
stage.addEventFilter(ActionEvent.ACTION,filter);
stage.addEventHandler(ActionEvent.ACTION,handler);
layout.addEventFilter(ActionEvent.ACTION, filter);
layout.addEventHandler(ActionEvent.ACTION,cutter);
button.addEventFilter(ActionEvent.ACTION, filter);
button.addEventHandler(ActionEvent.ACTION,cutter);
```

```
FILTER:STAGE javafx.stage.Stage@7c1031ba ==> Button[id=BUTTON-1, styleClass=button]
FILTER:SCENE javafx.scene.Scene@b30e9f8 ==> Button[id=BUTTON-1, styleClass=button]
FILTER:STACKPANE TilePane[id=STACKPANE, styleClass=root] ==> Button[id=BUTTON-1, styleClass=button]
FILTER:BUTTON-1 Button[id=BUTTON-1, styleClass=button] ==> Button[id=BUTTON-1, styleClass=button]
CUTTER:BUTTON-1 Button[id=BUTTON-1, styleClass=button] ==> Button[id=BUTTON-1, styleClass=button]
```